

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: MANAGEMENT OF INVALID TRACKS

APPLICANT: AMNON NAAMAD, YECHIEL YOCHAI AND SACHIN
MORE

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL486016425US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

Date of Deposit

2-25-02

Signature

Henry Jenkins

Typed or Printed Name of Person Signing Certificate

Henry Jenkins

MANAGEMENT OF INVALID TRACKS

FIELD OF INVENTION

This invention relates to data-storage systems, and in particular, to the management of invalid tracks in a data-storage system.

BACKGROUND

In a data-storage system, data is stored in logical volumes that are divided into cylinders, each of which includes several constituent tracks. In some cases, a data error may cause data on a particular track to become invalid. Whenever a track becomes invalid, it must be marked as such, both to prevent the data-storage system from relying on any data in that track and to draw attention to that track so that the data contained therein can be corrected or replaced.

A conventional data-storage system maintains a track table in which there exists an entry for each track. Each track table entry includes an invalid bit. A set invalid bit indicates that data in the corresponding track is invalid. Conversely, a clear invalid bit indicates that data in the corresponding track is valid. In normal operation, only a few isolated track table entries have a set invalid bit; the overwhelming majority of track table entries have clear invalid bits.

A repair process executing as a background task periodically scans the track table to identify any track table entries having set invalid bits. When the repair process encounters such an entry, it initiates corrective action, the nature of which depends on details of the configuration of the data-storage system.

In another application, a data-storage system configured to mirror data uses the invalid bit to trigger a mirroring event in which data is copied from a primary storage location to a mirror device. In such a system, whenever a change is made to a track, the invalid bit for that track is set. A copy process periodically scans the track table to identify any track table entries having invalid bits. If the copy process encounters such a track table entry, it recognizes the track as having data that must be copied to the mirror device.

Since track table entries having invalid bits are so few and far between, the repair process spends an inordinate amount of its time scanning the track table looking for

something to repair. This is an inefficient use of system resources. In addition, this lengthens the time during which a track holds invalid data as well as the time during which data on the primary storage location and on its mirror are different.

SUMMARY

The invention is based on the recognition that savings in time and system resources can be achieved by distilling status information concerning elementary data storage elements into status information concerning a data storage unit that includes two or more such data storage elements. In the implementation described herein, the data storage unit is the cylinder and the data storage element is a track. However, any grouping of data storage elements into data storage units is within the scope of the invention.

A method for implementing the invention is carried out in a data-storage system having a data storage unit that includes at least two constituent data storage elements. In one practice, the data storage unit corresponds to a cylinder and the data storage elements correspond to constituent tracks of the cylinder. However, any convenient element of data can be grouped into a larger data storage unit that includes several of those elements.

Each of the constituent data storage elements is either in a first state or a second state. The method includes providing a data structure having an entry corresponding to the data storage unit. The entry includes status information indicating whether at least one constituent data storage element of the data storage unit is in the first state. These entries are updated as necessary following any changes in state of the constituent data storage element.

In one practice of the invention, the data storage element is in its first state when there is invalid data stored therein. However, the invention does not depend on the meanings assigned to the first state and second states. Nor does it require that there only be two possible states.

In one practice of the invention, updating the entry includes identifying an entry in the data structure corresponding to a data storage unit that includes a constituent data storage element in its first state, and modifying status information in that entry to indicate that the data storage unit includes at least one constituent data storage element in the first

state. To avoid race conditions, the data structure is optionally locked before modifying status information and unlocked after modifying status information.

In another practice of the invention, modifying status information includes inspecting the status information to determine if it already indicates that at least one constituent data storage element is in the first state.

In another practice of the invention, updating the entry includes, after detecting that a constituent data storage element is in the second state, determining whether the data storage unit contains any constituent data storage element in the first state, identifying an entry in the data structure corresponding to a data storage unit that includes the constituent data storage element, and modifying status information in the entry to indicate that no constituent data storage elements of the data storage unit are in the first state.

In one practice of the invention, providing a data structure includes providing a bit map having a plurality of bits, each of which corresponds to a cylinder. Each bit has a first state and a second state. The first state indicates that there exists at least one invalid track in the cylinder. The second state indicates that there are no invalid tracks in the cylinder.

One practice of the invention includes scanning the data structure to locate constituent data storage elements in the first state. In one implementation, scanning the data structure includes detecting an entry in the data structure that indicates the presence, in the data storage unit associated with the data structure, of at least one constituent data storage element in the first state. Following such detection, this implementation includes scanning constituent data storage elements included in the data storage unit to identify the constituent data storage element in the first state.

These and other features and advantages of the invention will be apparent from the following detailed description and the accompanying figures, in which:

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 shows a data-storage system having a bitmap maintained in a shared memory;

FIG. 2 is a flow-chart of a process for setting a cylinder bit in the shared memory of

FIG. 1 following identification of an invalid track;

FIG. 3 is a flow-chart of a process for selectively clearing a cylinder bit from the shared memory of FIG. 1;

FIG. 4 is a flow-chart of a process for scanning the bitmap of FIG. 1 to identify cylinders with invalid-track entries; and

FIGS. 5-7 are pseudocoded implementations of the methods illustrated in FIGS. 2-4.

DETAILED DESCRIPTION

Referring to FIG. 1, a data-storage system **10** for serving the storage needs of several hosts **12A-Q** includes several host adaptors **14A-Q**, each of which is in communication with a host **12A-12Q**, and several disk adaptors **16A-Y**, each of which controls a disk subsystem **18A-Y** on which data is stored. Each disk subsystem **18A-Y** includes one or more disks, each of which is connected to a SCSI bus associated with a disk adaptor **16A-Y**. Each host adaptor **14A** and each disk adaptor **16A** includes its own processor and local memory. Hence, the host adaptors **14A-Q** and disk adaptors **16A-Y** can be viewed as independent processing elements.

The host adaptors **14A-Q** and the disk adaptors **16A-Y** share one or more buses **20** for communicating with a global memory **22**. By caching data in global memory **22**, the data-storage system **10** postpones high latency operations, such as disk access, and thereby enhances a host's perception of performance. The global memory **22** also includes metadata **24** used for managing the operation of the data-storage system **10**. This metadata **24** is shared among the host adaptors **14A-Q** and the disk adaptors **16A-Y**.

Included in this metadata **24** is a track table **26**, depicted in FIG. 2. The track table **26** contains an entry, hereafter referred to as a "track-table entry **28**", for each track on each disk in the data-storage system **10**. The track-table entry **28** for a particular track indicates, among other things, whether a copy of that track exists in the global memory **22**, and whether a copy of the track in global memory **22** needs to be copied to the disk that contains that track.

In some cases, data contained in a track can become corrupted. Corrupted data can be detected by, for example detecting an inconsistency in the CRC data for that track. When corrupted data is detected in a track, the track is flagged. For this purpose, each track-table entry **28** includes an invalid bit **30** that is set when the corresponding track includes corrupted data. A track-table entry **29** having a set invalid bit **30** shall be referred to herein as an "invalid-track entry **29**". The track that corresponds to an invalid-track entry **29** shall be referred to herein as an "invalid track".

Each disk adaptor **16A** periodically executes a repair utility as a background task. The repair utility searches for invalid tracks and, upon encountering an invalid track, performs some action. In doing so, the repair utility must access the track table **26** to search for invalid-track entries **29**. Each access to the track table **26** requires access to global memory **22**, which in turn requires the disk adaptor **16A** to contend for bus access. To the extent that the disk adaptor **16A** uses a bus **20** to inspect the track table **26** in global memory **22**, other processes cannot use that bus **20**. Because the disk adaptors **16A** and the host adaptors **14A** all contend for bus access, and because there are many fewer buses **20** than there are either disk adaptors **16A** or host adaptors **14A**, it is likely that a disk adaptor **16A** will have to wait for bus access.

In normal operation, there are very few invalid tracks. As a result, any invalid-track entries **29** in the track table **26** are few and far between. To enable the repair utility to identify invalid-track entries **29** with fewer accesses to global memory **22**, the data-storage system **10** maintains a data structure **32** having a plurality of elements, each of which corresponds to a set of tracks. Since each element corresponds to a set of tracks, the number of elements in the data structure **32** is less than the number of track-table entries.

Each element of the data structure **32** can be in one of two states. An element is in its first state only when there exists at least one invalid track within the set of tracks identified by that element. An element is in its second state when there are no invalid tracks in the set of tracks identified by that element.

In one embodiment, the set of tracks identified by an element in the data structure **32** is a contiguous set of tracks. A convenient choice for a set of tracks is the set of fifteen contiguous tracks referred to as a cylinder on a disk.

A suitable data structure **32** is a bitmap **40** in which each constituent bit, hereafter referred to as a cylinder bit **42**, represents one cylinder. A cylinder bit **42** is set if its corresponding cylinder includes at least one invalid track. In all other cases, the cylinder bit **42** is clear.

Maintenance of the bitmap requires that a cylinder bit be set whenever a track in that cylinder contains invalid data. FIG. 2 shows a procedure to be used for updating the track-table entries whenever an invalid bit is set. Following the setting of an invalid bit associated with a track (step **52**), the cylinder bit that corresponding to that track is identified (step **54**). The bitmap is then locked to prevent any race condition from occurring (step **56**) and the appropriate cylinder bit is set if necessary (step **58**). Finally the bitmap is unlocked (step **62**). The procedure then stops after the last track-table entry has been processed (step **62**).

Maintenance of the bitmap also requires that a cylinder bit be cleared once all tracks on the cylinder bit's associated cylinder contain valid data. However, before a repair utility can clear a cylinder bit, it must confirm that there are no more invalid tracks in the corresponding cylinder. This is because a set cylinder bit indicates that there exists at least one invalid track in the cylinder. It does not indicate how many invalid tracks there might be in that cylinder.

FIG. 3 shows the process carried out by the repair utility after having cleared the invalid bit in a particular track-table entry (step **64**). The cylinder that contains the track whose invalid bit is to be cleared is identified (step **66**) and the bitmap is locked to prevent any race condition from occurring (step **68**). The invalid bits for all track-table entries in that portion of the track table that corresponds to that cylinder are then inspected (step **70**). If no set invalid bits are detected in that portion of the track table (step **72**), the cylinder bit is cleared (step **74**) and the bitmap is unlocked (step **76**). If a set invalid bit is detected (step **72**), the cylinder bit remains set and the bitmap is unlocked (step **76**). Once the bitmap is unlocked, the procedure stops (step **78**).

The foregoing maintenance steps enable the data-storage to identify invalid tracks with fewer accesses to the shared memory. This benefit results from the distillation, into one cylinder bit in the bitmap, of status information concerning all tracks in a cylinder. As a

result, the entire track table need not be scanned to identify invalid tracks. Instead, the much smaller bitmap is scanned, and only a limited portion of the track table need is scanned when a set cylinder bit is encountered in the bitmap.

FIG. 4 shows the manner in which the data-storage system identifies invalid tracks. The cylinder bits in the bitmap are periodically inspected (step 84) to determine whether there are any set cylinder bits (step 86). If a cylinder bit is clear, the next cylinder bit is inspected (step 84). Otherwise, that portion of the track table that corresponds to the cylinder represented by the cylinder bit is scanned to identify the particular invalid track (step 88). If an invalid-entry is found (step 90). When it does, the invalid data is corrected (step 92) and the cylinder bit is either cleared or not depending on the outcome of the procedure described in connection with FIG. 3 (step 94). In rare cases, a cylinder bit is set but no invalid-track entries are found (step 90). When this is the case, an error is posted (step 96).

FIG. 5 shows a particular algorithm, written as pseudocode, for updating the bitmap when an invalid track is identified. The cylinder bit corresponding to the invalid track is first identified (step 100). The integer "X" is a pointer to the first byte of the bitmap. The integer "O" is the offset, relative to the first byte of the bitmap, of the byte that contains the cylinder bit containing the invalid track. The integer "B" is the offset of the cylinder bit within that byte.

Once the byte containing the cylinder bit is identified, the bitmap is locked (step 102) so that other processes cannot modify it. Then, the byte that contains the cylinder bit is read into the integer D (step 104). If the cylinder bit indicates that there are no invalid tracks on the cylinder clear, it is changed to indicate that there is at least one invalid track in the cylinder (step 106). Then, the byte is written back into the bitmap (step 108) and the bitmap is unlocked (step 110). Otherwise, the cylinder bit is already set, in which case there is no need to write anything to the bitmap. In this case, the bitmap is simply unlocked (step 110).

FIG. 6 shows another algorithm for updating the bitmap, in this case after an invalid track (hereafter referred to as the "restored track") has been restored to being a valid track. In FIG. 6, the location of the cylinder bit that contains the restored track is identified in the

same manner as in FIG. 5 (step 112). Once the byte containing the cylinder bit is identified, the bitmap is locked (step 114) so that other processes cannot modify it. Then, the byte that contains the cylinder bit is read into the integer D (step 116).

In this case, a set cylinder bit cannot be cleared until all tracks containing that are mapped to that cylinder bit are confirmed to be valid tracks. The illustrated procedure does this by setting a counter equal to the number of tracks in the cylinder (step 117), inspecting each track (step 118), and decrementing the counter each time a valid track is encountered (step 119). If at the end of this, the counter is zero, then all tracks on the cylinder must be valid (step 120), in which case the cylinder bit is cleared (step 121), the byte containing the cylinder bit is written back into the bitmap (step 122), and the bitmap is unlocked (step 123). If the cylinder bit is already clear going into this procedure, then it follows that there should have been no invalid tracks in the cylinder. Under these circumstances, it is troubling that this procedure, which is called only after an invalid track is restored, was ever called to begin with. Because of this inconsistency, the procedure posts an error (step 124).

FIG. 7 shows an algorithm for identifying invalid tracks. The process begins with identifying the next track following the most recently identified invalid track (step 126). The byte and offset for the cylinder bit associated with the next track is then identified in the same manner as discussed in connection with FIG. 5 (step 128).

If the entire byte containing the cylinder bit is clear, then there are no invalid tracks in any of the eight cylinders encompassed by the cylinder bits in that byte. In that case, the procedure skips over all 120 (8×15) tracks mapped to the eight bits in that byte (step 129).

However, if the byte includes at least one non-zero cylinder bit (step 130), the cylinder bits in that byte are inspected to identify the particular cylinder having the invalid track (step 132). Each track-table entry for that particular cylinder is then inspected to determine whether it is an invalid-track entry (step 134). The track number for the first invalid-track entry encountered is returned by the procedure (step 136). The procedure then moves to the next track (step 138). Note that since all fifteen tracks in the particular cylinder have just been inspected, the next track is actually offset by the number of tracks mapped to a cylinder bit (fifteen).

In the embodiment described herein, status information concerning tracks is distilled by grouping the tracks by cylinder and setting a bit corresponding to each cylinder. However, other groupings are possible. For example, if invalid tracks occur more frequently, it may be useful to assign less than a cylinder's worth of tracks to each bit in the bitmap. If invalid tracks occur less frequently, it may be useful to assign more than one cylinder's worth of tracks to each bit.

Additionally, the units of data storage need not be cylinders and tracks. For example, the bits in the bitmap may represent collections of blocks and the bits would then be set if any block in the collection of blocks is invalid.

More generally, the method described herein achieves savings in time and system resources by grouping elementary data storage elements into data storage units that include two or more such data storage elements. In the implementation described herein, the data storage unit is the cylinder and the data storage element is a track. However, any grouping of data storage elements into data storage units is within the scope of the invention.

The method described herein uses two layers of a data storage hierarchy. However, there is no reason the method cannot be extended to include multiple layers of a data storage hierarchy. For example, there may be a second bitmap in which each bit corresponds to one byte of the bitmap shown in FIG. 1. This second bitmap would then be scanned first. If a set bit is found in the second bitmap, the corresponding byte from the bitmap shown in FIG. 1 is scanned until the set cylinder bit is found, as already described in connection with FIG. 3. An implementation of this type is also within the scope of the invention.

Having described the invention, and a preferred embodiment thereof, what we claim as new and secured by letters patent is: